INDIAN PREMIER LEAGUE ®

1st Inning Score Predictor

ST 3082 - Final Project

Group 4
M. D. D. De Costa – s14533
S. P. P. M. Sudasinghe – s14510
K. G. S. K. Wickramasinghe - s14594

Abstract

This report is based on data recorded in the Indian Premier League (IPL) cricket matches in the years 2008-2017. The objective of this analysis is to develop a predictive model that predicts the total score of the team batting first, based on the current state of the match (during 5th to 20th over). The software package that we used to do this analysis was Python. We deployed many machine learning models such as Linear, Ridge, Lasso, Elastic Net Regression and also Random Forest Model. Also, we ran a deep learning model, Artificial Neural Network. Using the Random Forest Model, we were able to get the most accurate model to predict the final total of the first inning in a given cricket match.

Table of Contents

List of Tables

List of Figures

Introduction

Indian Premier League (IPL) is a professional men's Twenty20 cricket league, contested by eight teams based out of eight Indian cities. This now major cricketing franchise event that attracts hundreds of million USD league was founded by the Board of Control for Cricket in India (BCCI) in 2007. The IPL is the most-attended cricket league in the world and in 2014 was ranked sixth by average attendance among all sports leagues. With this much interest, inevitably, hundreds of millions of USD gets passed around from this event in legitimate and illegitimate ways.

Objective

Our main objective of the analysis is to predict the total that the team playing first is going to get, after only 5 overs have been played. We do this by modelling various variables using past data. Apart, to identify the impact of various factors that affect the 1st inning total.

About the dataset

We collected the dataset from Kaggle repository. The dataset contains ball by ball data of the 1st inning of each IPL match that was played from 2008 to 2017. The raw data contains 76014 rows and 15 columns.

Variable	**Description**
mid	Match ID
date	Date the match was played
venue	The ground the match was played in
batting_team	Team batting first
bowling_team	Team bowling first
batsman	The batsman facing the specific delivery
bowler	The bowler who is bowling the specific delivery
runs	Runs conceived in that specific delivery
wickets	Total wickets conceived right after that delivery
overs	Total overs and bowls bowled right after that delivery
runs_last_5	Total runs conceived in the last 5 overs before that ball
wickets_last_5	Total wickets conceived in the last 5 overs before that ball
striker	Runs scored by the striker of the ball
non.striker	Runs scored by the non-striker of the ball
total	Total runs conceived in that inning

Table 1

Data Cleaning

- Looking at the data we realized that there were many columns that did not give any meaningful contribution to the total value. So first, we removed those few columns. Those columns were, 'date','mid','batsman','bowler','striker','non-striker','venue'.
- Then, we realized that there were some team name changes in between some seasons. They were same franchise teams with the same owners but only the team’s name was different. So, we combined all the records with before and after the team’s name change as a single team. Those name changes were, ‘Deccan Chargers’ were renamed as ‘Sunrises Hyderabad’ in 2012. Also, ‘Delhi Daredevils’ were renamed as ‘Delhi Capitals’ in 2018.
- There were 4 teams that played in some of the seasons of the IPL but were now playing now. So, we removed all the records containing those specific teams. Those teams that we removed were, 'Pune Warriors', 'Kochi Tuskers Kerala', 'Rising Pune Supergiants', 'Gujarat Lions'.
- Finally, since we can’t naturally get a good prediction about the final total at least some of the overs were played in the match, we decided to remove the records of the first 5 overs of the first inning. Our decision here was also influenced by the fact that the dataset contained the columns, ‘runs_last_5’ and ‘wickets_last_5’.

Exploratory Data Analysis

Since the target variable, 'Total, is the most important variable here, it is good to understand its distribution. According to the following density plot, the distribution of the variable 'Total' is approximately normally distributed.

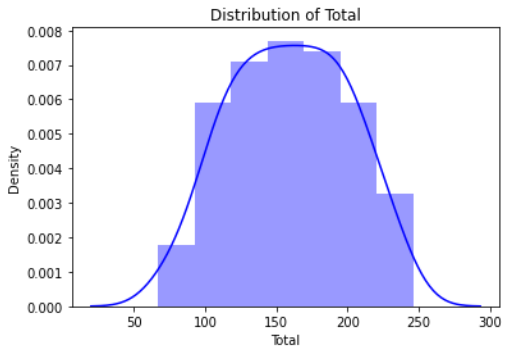


Figure 1

According to this table, team Chennai Super Kings scored the highest 1st inning score (246) against Rajasthan Royals. The second-highest total runs were obtained by them also. When considering the top 5 first inning totals, Kings Xi Punjab also seems to be a team that gives opponents a good target.

Batting_team	**Bowling_team**	**total**
Chennai Super Kings	Rajasthan Royals	246
Chennai Super Kings	Kings XI Punjab	240
Royal Challengers Bangalore	Mumbai Indians	235
Kings XI Punjab	Royal Challengers Bangalore	232
Kings XI Punjab	Chennai Super Kings	231

Table 2

From the boxplot below, we can conclude that, on average, team Chennai Super Kings sets a good target against the bowling team. According to the records shown in this plot, if team Delhi Capitals have to bat first, they score, on average, fewer than other teams. As mentioned earlier, this plot also implies that team Chennai Super Kings was the highest total setter in the previous matches. Royal Challengers Bangalore and Kolkata Knight Riders scored the lower 1st inning totals.

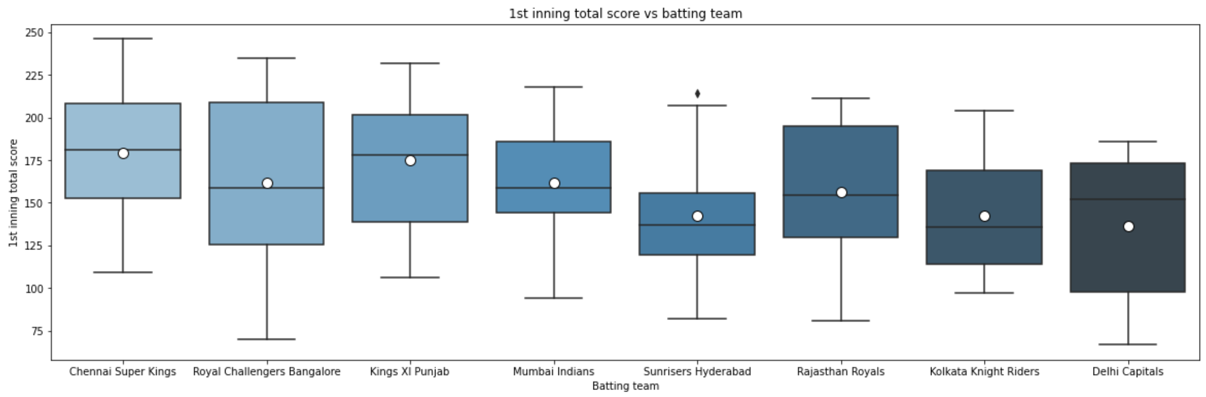


Figure 2

From the boxplot below, we can conclude that, on average, other teams set a good target against the team Sunrises Hyderabad. It seems that team Royal Challengers Bangalore bowl with more control against the 1st inning batting team. As mentioned earlier, this boxplot also confirmed that Rajasthan Royals had to chase the highest target.

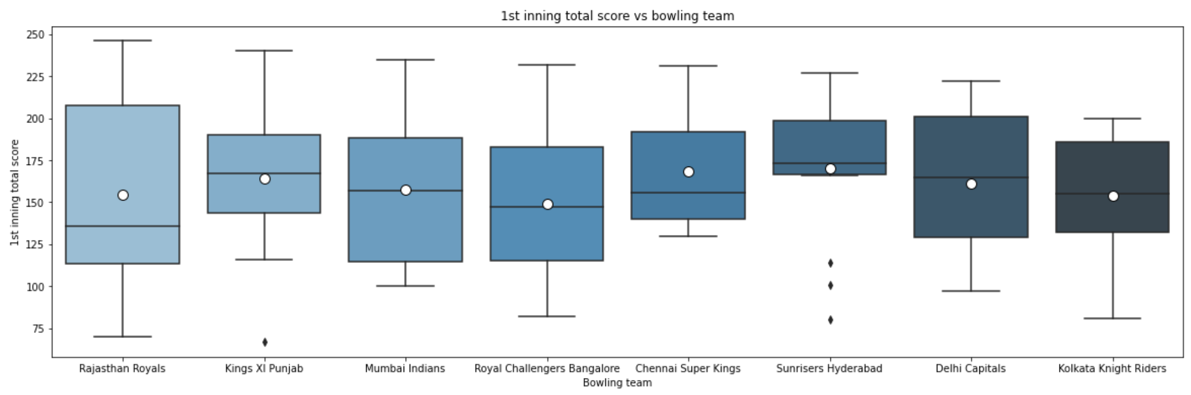


Figure 3

Suggestions for advanced analysis

Univariate and bivariate plots between the response and predictor variables led the pathway to identify the different techniques that could be used for the advanced analysis. Spearman rank correlation test was performed to get an idea about the association between continuous predictor variables. As you can see in the graph below, there seems to be some heavy correlations between some of the continuous variables. For example, the runs obtained in the last 5 overs is highly correlated with the total score.

So, for the advanced analysis we will be using linear, ridge, lasso, elastic net and random forest regression. Also, to check whether we can further improve the accuracy, we will be fitting a neural network to the data as well.

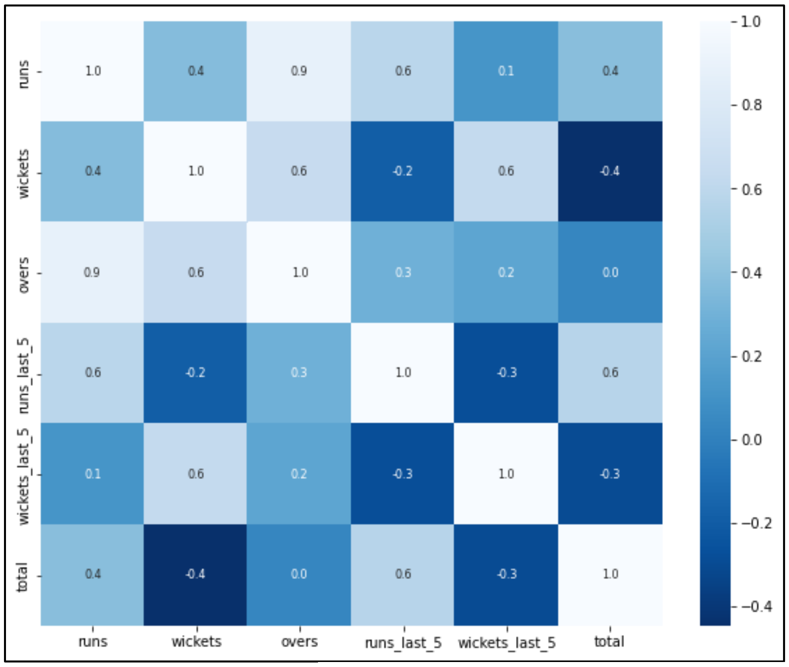


Figure 4

Important Results of the Advanced Analysis

With the objective of predicting the 1st inning score of IPL, a linear regression, ridge, lasso and elastic net were fitted to the dataset. Further to reduce test RMSE, random forest and neural network models were fitted and obtained the models with the highest accuracy and lowest test RMSE.

Linear Regression Model

To predict the score, basic model linear regression was used. The obtained MAE, MAPE%, RMSE values are given in the table.

	MAE	MAPE%	RMSE
Train set	12.939	8.603	17.234
Test set	12.866	8.485	17.015

Table 3

Ridge Regression

5-fold cross validation was used to select the best parameter alpha which is 40. Best score given by the ridge was -297.3588 under negative mean squared error.

	MAE	MAPE%	RMSE
Train set	12.937	8.602	17.234
Test set	12.863	8.484	17.014

Table 4

Lasso Regression

Here also 5-fold cross validation was used to select the best parameter alpha and it gives 0.001. Best score given by the model was -297.3595.

	MAE	MAPE%	RMSE
Train set	12.938	8.603	17.234
Test set	12.865	8.485	17.014

Table 5

Elastic Net Regression

The best alpha corresponding to the 5-fold cross validation was 0.001 and the best score given by this model was -297.3595.

	MAE	MAPE%	RMSE
Train set	12.938	8.602	17.234
Test set	12.864	8.485	17.014

Table 6

Random Forest Regression

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. Therefore, to check whether the test RMSE can be reduced, the Random Forest was fitted. We used 1000 trees in the model and minimum number of samples required to be at a leaf node was 2.

	MAE	MAPE%	RMSE
Train set	3.392	2.198	5.277
Test set	5.712	3.653	8.591

Table 7

Neural Network Model

The neural network that we built comprises of an input layer, one hidden layer with 100 neurons, and an output layer. Logistic was used as the activation function for the hidden layer and the 500 iterations was used as the maximum number of iterations.

	MAE	MAPE%	RMSE
Train set	8.119	5.253	11.254
Test set	8.418	5.413	11.519

Table 8

Comparison of all models

By comparing the results obtained from above models, it was observed that Random Forest Model gives the lowest Test RMSE. Also, the MAPE value of 3.653% which is a good value. Therefore, we choose the Random Forest Model as our best model and use that model to build our data product.

	RMSE	
	Train	Test
Linear Regression	17.234333	17.014622
Ridge Regression	17.234355	17.013937
Lasso Regression	17.234337	17.014486
Elastic Net Regression	17.234342	17.014237
Random Forest	5.277110	8.590519
Neural Network	11.253649	11.519192

Table 9

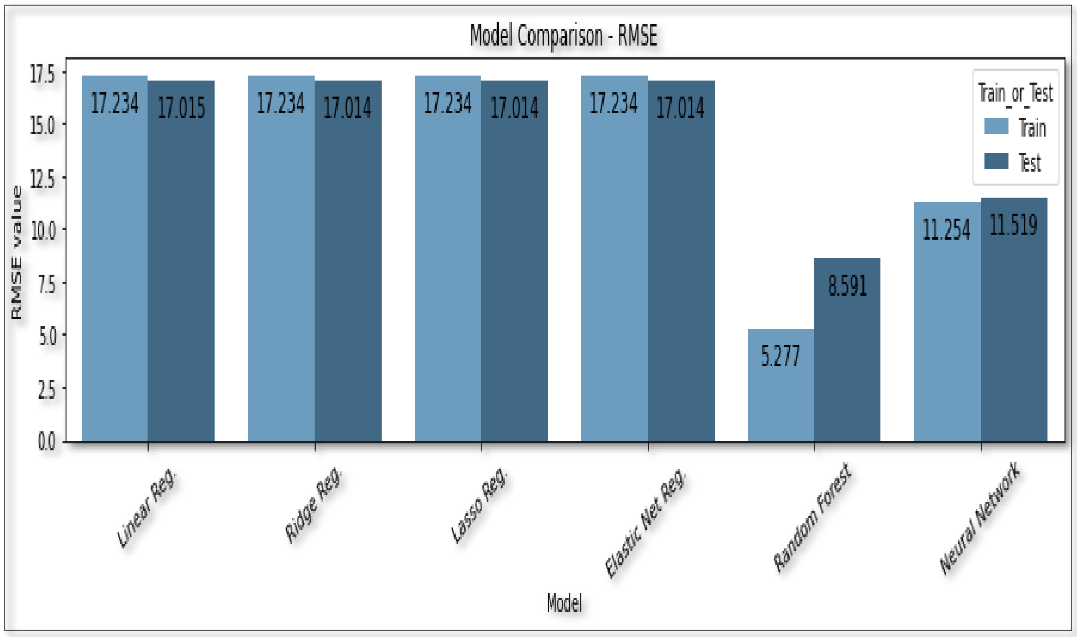


Figure 5

Issues encountered and proposed solutions

- There was a considerable overfitting problem in our random forest model. Thus, hyperparameter tunning was done by changing some parameters to overcome this problem.
- We noticed that the final score (Total) of each match was repeated in every ball, which was played by that team. So, we had to take unique totals for the descriptive analysis.
- Initially we ran the models in Jupyter Notebook but it took very long time to run models such as Random Forest, Neural Network. So, we used Google Collaboratory which decreased the time taken significantly.

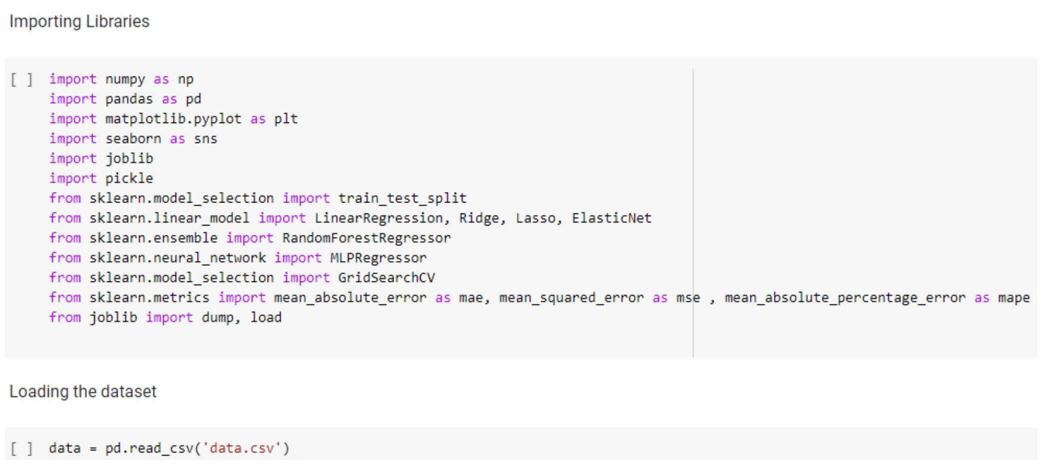
Discussion and Conclusion

This study was conducted to predict the total that the team playing first is going to get and to identify the impact of various factors that affect the 1st inning score.

According to the initial descriptive analysis our target variable 'Total' was approximately normally distributed. And we saw that team Chennai Super Kings scored the highest 1st inning score and from the boxplot also we saw that Chennai Super Kings sets a good target against the bowling team.

Then the Linear regression, ridge, lasso, elastic net models were fitted and further to reduce test RMSE random forest and neural network models were fitted. Among these models Random Forest Model gave the lowest Test RMSE (8.590519) and lowest test MAPE (3.653%). Therefore, we selected this model as the best model. Furthermore, this model will be used to develop the data product application which can be used to get the predictions of 1st inning score. The data product will be developed using Python.

Python Code

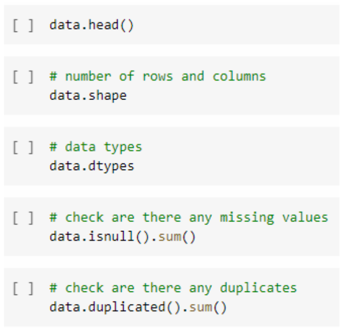


Importing Libraries

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import joblib
     import pickle
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
     from sklearn.ensemble import RandomForestRegressor
     from sklearn.neural_network import MLPRegressor
     from sklearn.model_selection import GridSearchCV
     from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as mse , mean_absolute_percentage_error as mape
     from joblib import dump, load
```

Loading the dataset

```
[ ]  data = pd.read_csv('data.csv')
```



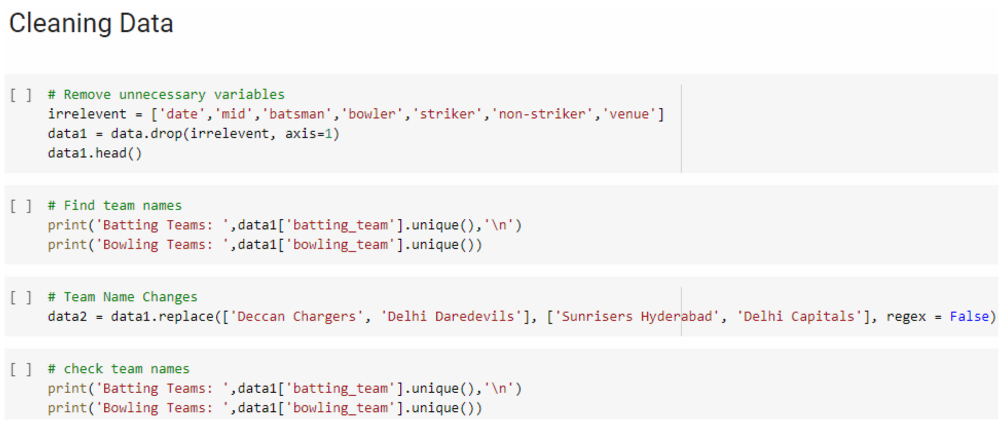
```
[ ] data.head()
```

```
[ ] # number of rows and columns
    data.shape
```

```
[ ] # data types
    data.dtypes
```

```
[ ] # check are there any missing values
    data.isnull().sum()
```

```
[ ] # check are there any duplicates
    data.duplicated().sum()
```



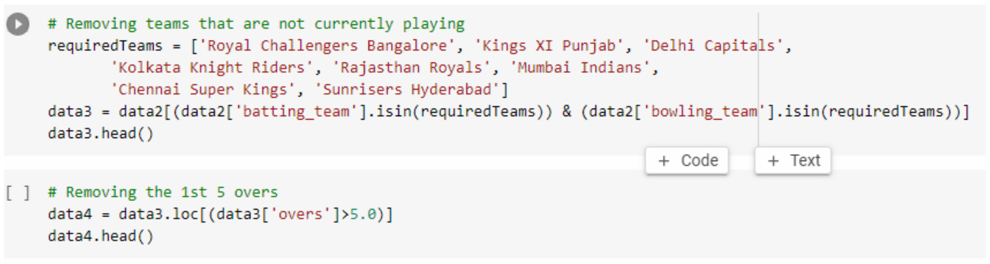
Cleaning Data

```
[ ] # Remove unnecessary variables
    irrelevent = ['date','mid','batsman','bowler','striker','non-striker','venue']
    data1 = data.drop(irrelevent, axis=1)
    data1.head()
```

```
[ ] # Find team names
    print('Batting Teams: ',data1['batting_team'].unique(),'\n')
    print('Bowling Teams: ',data1['bowling_team'].unique())
```

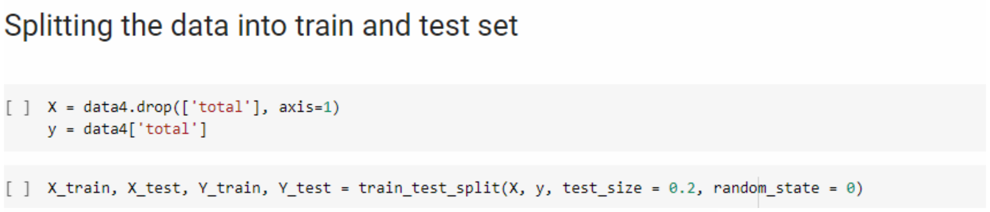
```
[ ] # Team Name Changes
    data2 = data1.replace(['Deccan Chargers', 'Delhi Daredevils'], ['Sunrisers Hyderabad', 'Delhi Capitals'], regex = False)
```

```
[ ] # check team names
    print('Batting Teams: ',data1['batting_team'].unique(),'\n')
    print('Bowling Teams: ',data1['bowling_team'].unique())
```



```
# Removing teams that are not currently playing
requiredTeams = ['Royal Challengers Bangalore', 'Kings XI Punjab', 'Delhi Capitals',
        'Kolkata Knight Riders', 'Rajasthan Royals', 'Mumbai Indians',
        'Chennai Super Kings', 'Sunrisers Hyderabad']
data3 = data2[(data2['batting_team'].isin(requiredTeams)) & (data2['bowling_team'].isin(requiredTeams))]
data3.head()
```

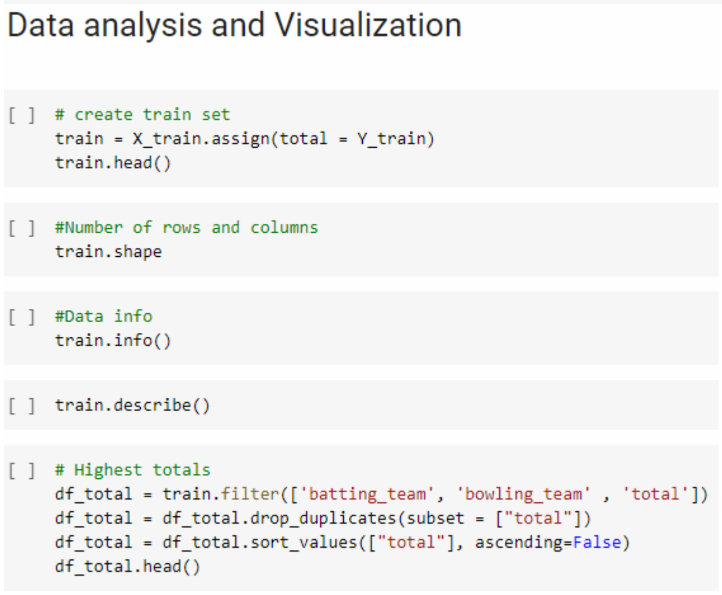
```
[ ] # Removing the 1st 5 overs
    data4 = data3.loc[(data3['overs']>5.0)]
    data4.head()
```



Splitting the data into train and test set

```
[ ] X = data4.drop(['total'], axis=1)
    y = data4['total']
```

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```



Data analysis and Visualization

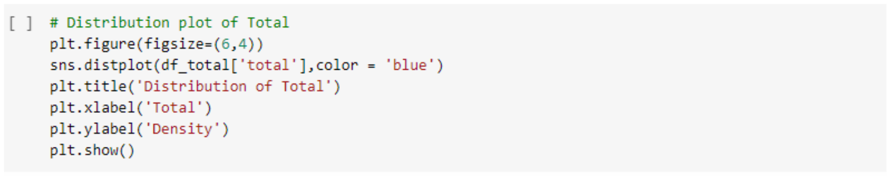
```
[ ] # create train set
    train = X_train.assign(total = Y_train)
    train.head()
```

```
[ ] #Number of rows and columns
    train.shape
```

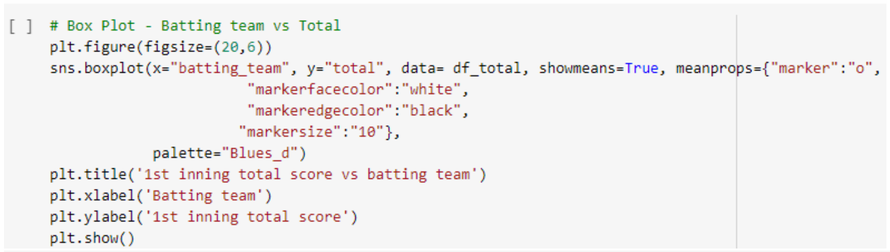
```
[ ] #Data info
    train.info()
```

```
[ ] train.describe()
```

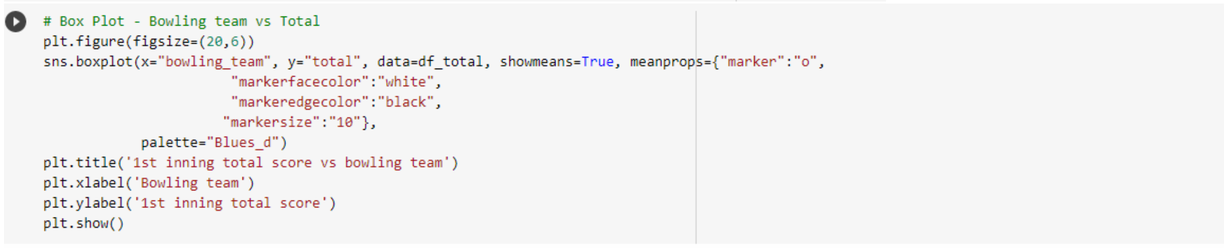
```
[ ] # Highest totals
    df_total = train.filter(['batting_team', 'bowling_team' , 'total'])
    df_total = df_total.drop_duplicates(subset = ["total"])
    df_total = df_total.sort_values(["total"], ascending=False)
    df_total.head()
```



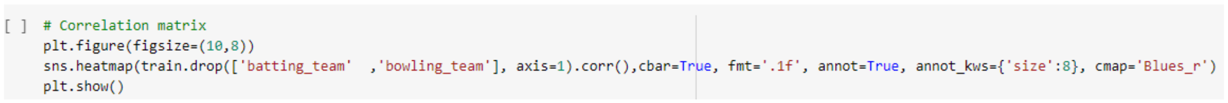
```
[ ]  # Distribution plot of Total
     plt.figure(figsize=(6,4))
     sns.distplot(df_total['total'],color = 'blue')
     plt.title('Distribution of Total')
     plt.xlabel('Total')
     plt.ylabel('Density')
     plt.show()
```



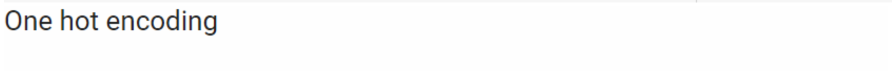
```
[ ]  # Box Plot - Batting team vs Total
     plt.figure(figsize=(20,6))
     sns.boxplot(x="batting_team", y="total", data= df_total, showmeans=True, meanprops={"marker":"o",
                            "markerfacecolor":"white",
                            "markeredgecolor":"black",
                           "markersize":"10"},
                 palette="Blues_d")
     plt.title('1st inning total score vs batting team')
     plt.xlabel('Batting team')
     plt.ylabel('1st inning total score')
     plt.show()
```



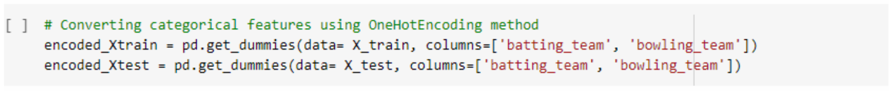
```
# Box Plot - Bowling team vs Total
plt.figure(figsize=(20,6))
sns.boxplot(x="bowling_team", y="total", data=df_total, showmeans=True, meanprops={"marker":"o",
                       "markerfacecolor":"white",
                       "markeredgecolor":"black",
                      "markersize":"10"},
            palette="Blues_d")
plt.title('1st inning total score vs bowling team')
plt.xlabel('Bowling team')
plt.ylabel('1st inning total score')
plt.show()
```



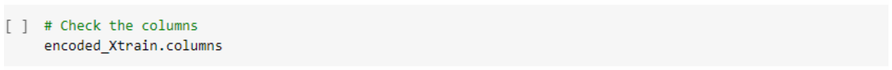
```
[ ]  # Correlation matrix
     plt.figure(figsize=(10,8))
     sns.heatmap(train.drop(['batting_team'  ,'bowling_team'], axis=1).corr(),cbar=True, fmt='.1f', annot=True, annot_kws={'size':8}, cmap='Blues_r')
     plt.show()
```



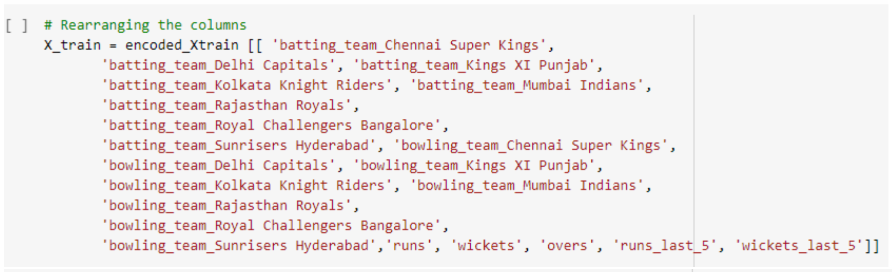
One hot encoding



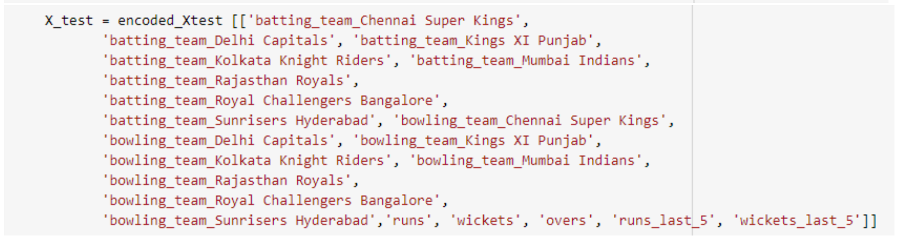
```
[ ]  # Converting categorical features using OneHotEncoding method
     encoded_Xtrain = pd.get_dummies(data= X_train, columns=['batting_team', 'bowling_team'])
     encoded_Xtest = pd.get_dummies(data= X_test, columns=['batting_team', 'bowling_team'])
```



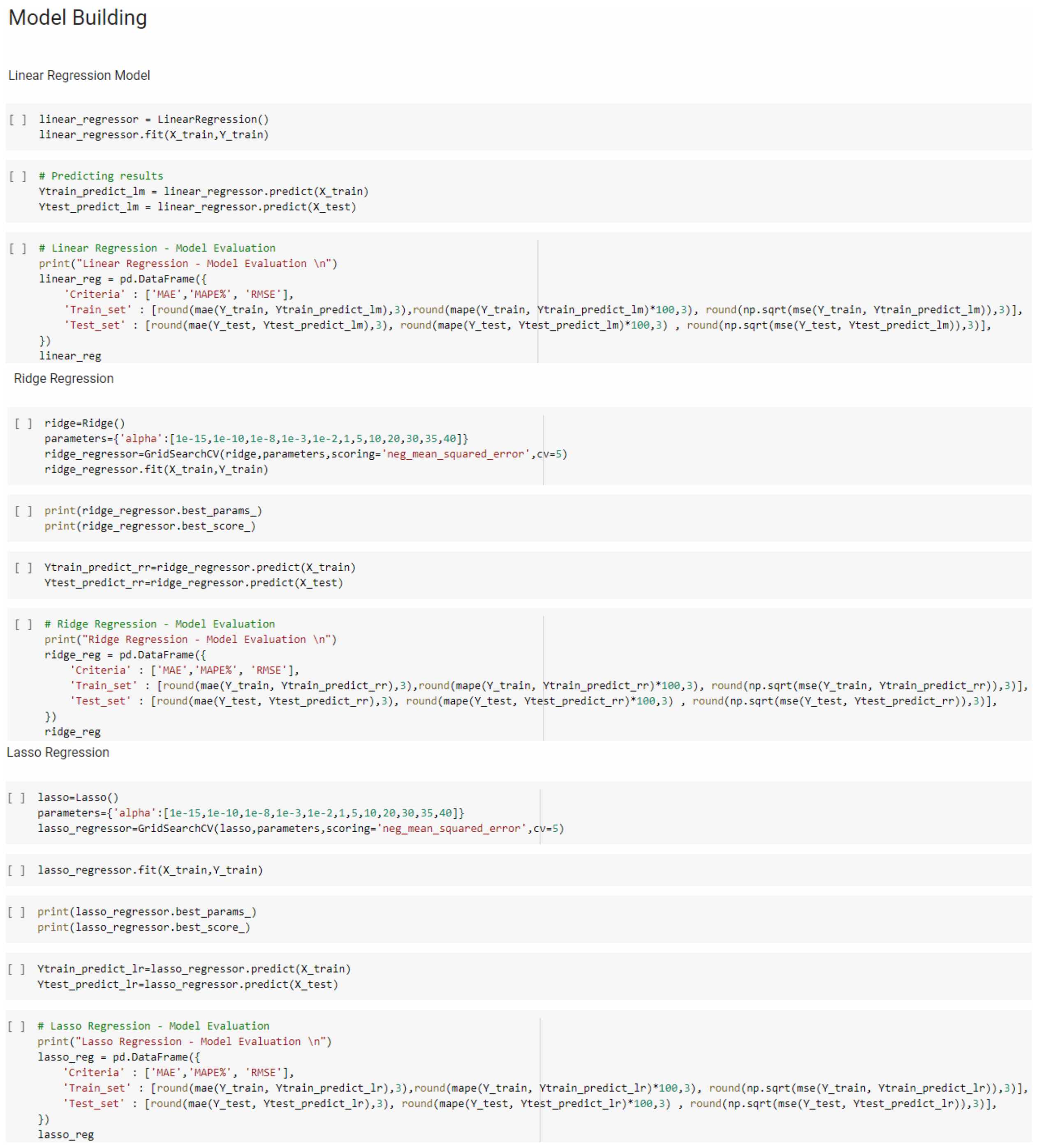
```
[ ]  # Check the columns
     encoded_Xtrain.columns
```



```
[ ]  # Rearranging the columns
     X_train = encoded_Xtrain [[ 'batting_team_Chennai Super Kings',
            'batting_team_Delhi Capitals', 'batting_team_Kings XI Punjab',
            'batting_team_Kolkata Knight Riders', 'batting_team_Mumbai Indians',
            'batting_team_Rajasthan Royals',
            'batting_team_Royal Challengers Bangalore',
            'batting_team_Sunrisers Hyderabad', 'bowling_team_Chennai Super Kings',
            'bowling_team_Delhi Capitals', 'bowling_team_Kings XI Punjab',
            'bowling_team_Kolkata Knight Riders', 'bowling_team_Mumbai Indians',
            'bowling_team_Rajasthan Royals',
            'bowling_team_Royal Challengers Bangalore',
            'bowling_team_Sunrisers Hyderabad','runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5']]
```



```
     X_test = encoded_Xtest [['batting_team_Chennai Super Kings',
            'batting_team_Delhi Capitals', 'batting_team_Kings XI Punjab',
            'batting_team_Kolkata Knight Riders', 'batting_team_Mumbai Indians',
            'batting_team_Rajasthan Royals',
            'batting_team_Royal Challengers Bangalore',
            'batting_team_Sunrisers Hyderabad', 'bowling_team_Chennai Super Kings',
            'bowling_team_Delhi Capitals', 'bowling_team_Kings XI Punjab',
            'bowling_team_Kolkata Knight Riders', 'bowling_team_Mumbai Indians',
            'bowling_team_Rajasthan Royals',
            'bowling_team_Royal Challengers Bangalore',
            'bowling_team_Sunrisers Hyderabad','runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5']]
```



Model Building

Linear Regression Model

```
[ ]  linear_regressor = LinearRegression()
     linear_regressor.fit(X_train,Y_train)
```

```
[ ]  # Predicting results
     Ytrain_predict_lm = linear_regressor.predict(X_train)
     Ytest_predict_lm = linear_regressor.predict(X_test)
```

```
[ ]  # Linear Regression - Model Evaluation
     print("Linear Regression - Model Evaluation \n")
     linear_reg = pd.DataFrame({
         'Criteria' : ['MAE','MAPE%', 'RMSE'],
         'Train_set' : [round(mae(Y_train, Ytrain_predict_lm),3),round(mape(Y_train, Ytrain_predict_lm)*100,3), round(np.sqrt(mse(Y_train, Ytrain_predict_lm)),3)],
         'Test_set' : [round(mae(Y_test, Ytest_predict_lm),3), round(mape(Y_test, Ytest_predict_lm)*100,3) , round(np.sqrt(mse(Y_test, Ytest_predict_lm)),3)],
     })
     linear_reg
```

Ridge Regression

```
[ ]  ridge=Ridge()
     parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40]}
     ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=5)
     ridge_regressor.fit(X_train,Y_train)
```

```
[ ]  print(ridge_regressor.best_params_)
     print(ridge_regressor.best_score_)
```

```
[ ]  Ytrain_predict_rr=ridge_regressor.predict(X_train)
     Ytest_predict_rr=ridge_regressor.predict(X_test)
```

```
[ ]  # Ridge Regression - Model Evaluation
     print("Ridge Regression - Model Evaluation \n")
     ridge_reg = pd.DataFrame({
         'Criteria' : ['MAE','MAPE%', 'RMSE'],
         'Train_set' : [round(mae(Y_train, Ytrain_predict_rr),3),round(mape(Y_train, Ytrain_predict_rr)*100,3), round(np.sqrt(mse(Y_train, Ytrain_predict_rr)),3)],
         'Test_set' : [round(mae(Y_test, Ytest_predict_rr),3), round(mape(Y_test, Ytest_predict_rr)*100,3) , round(np.sqrt(mse(Y_test, Ytest_predict_rr)),3)],
     })
     ridge_reg
```

Lasso Regression

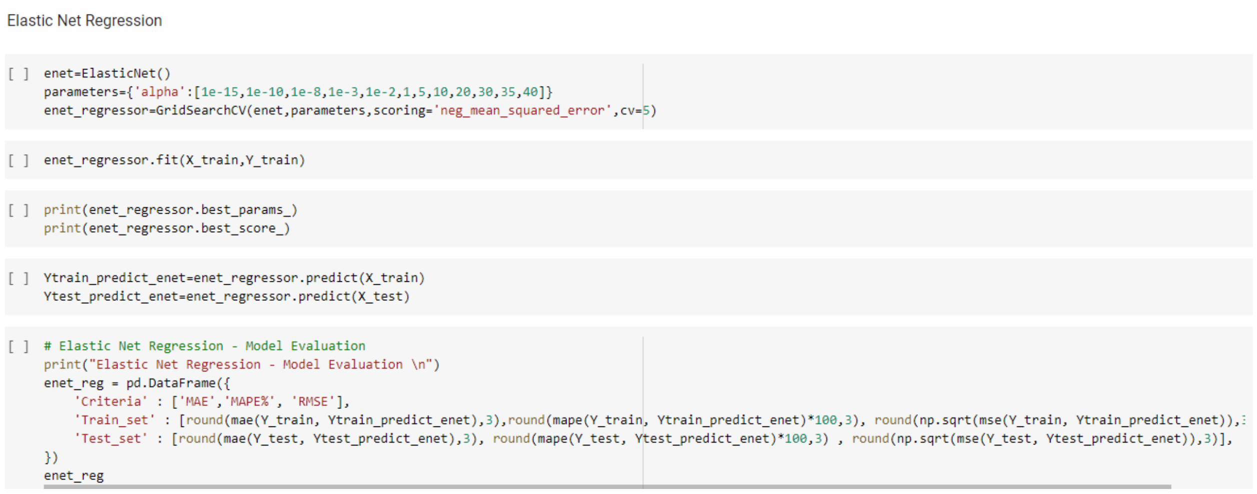
```
[ ]  lasso=Lasso()
     parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40]}
     lasso_regressor=GridSearchCV(lasso,parameters,scoring='neg_mean_squared_error',cv=5)
```

```
[ ]  lasso_regressor.fit(X_train,Y_train)
```

```
[ ]  print(lasso_regressor.best_params_)
     print(lasso_regressor.best_score_)
```

```
[ ]  Ytrain_predict_lr=lasso_regressor.predict(X_train)
     Ytest_predict_lr=lasso_regressor.predict(X_test)
```

```
[ ]  # Lasso Regression - Model Evaluation
     print("Lasso Regression - Model Evaluation \n")
     lasso_reg = pd.DataFrame({
         'Criteria' : ['MAE','MAPE%', 'RMSE'],
         'Train_set' : [round(mae(Y_train, Ytrain_predict_lr),3),round(mape(Y_train, Ytrain_predict_lr)*100,3), round(np.sqrt(mse(Y_train, Ytrain_predict_lr)),3)],
         'Test_set' : [round(mae(Y_test, Ytest_predict_lr),3), round(mape(Y_test, Ytest_predict_lr)*100,3) , round(np.sqrt(mse(Y_test, Ytest_predict_lr)),3)],
     })
     lasso_reg
```



Elastic Net Regression

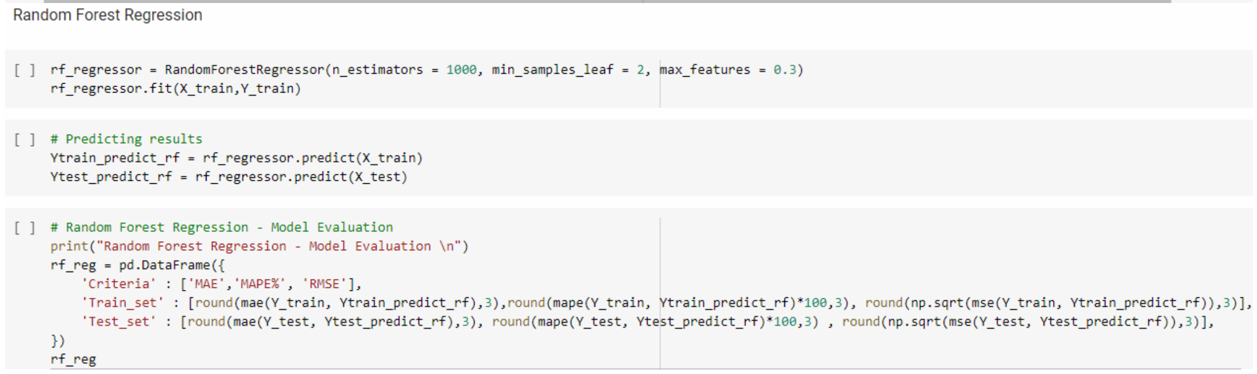
```
[ ]  enet=ElasticNet()
     parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40]}
     enet_regressor=GridSearchCV(enet,parameters,scoring='neg_mean_squared_error',cv=5)
```

```
[ ]  enet_regressor.fit(X_train,Y_train)
```

```
[ ]  print(enet_regressor.best_params_)
     print(enet_regressor.best_score_)
```

```
[ ]  Ytrain_predict_enet=enet_regressor.predict(X_train)
     Ytest_predict_enet=enet_regressor.predict(X_test)
```

```
[ ]  # Elastic Net Regression - Model Evaluation
     print("Elastic Net Regression - Model Evaluation \n")
     enet_reg = pd.DataFrame({
         'Criteria' : ['MAE','MAPE%', 'RMSE'],
         'Train_set' : [round(mae(Y_train, Ytrain_predict_enet),3),round(mape(Y_train, Ytrain_predict_enet)*100,3), round(np.sqrt(mse(Y_train, Ytrain_predict_enet)),:
         'Test_set' : [round(mae(Y_test, Ytest_predict_enet),3), round(mape(Y_test, Ytest_predict_enet)*100,3) , round(np.sqrt(mse(Y_test, Ytest_predict_enet)),3)],
     })
     enet_reg
```

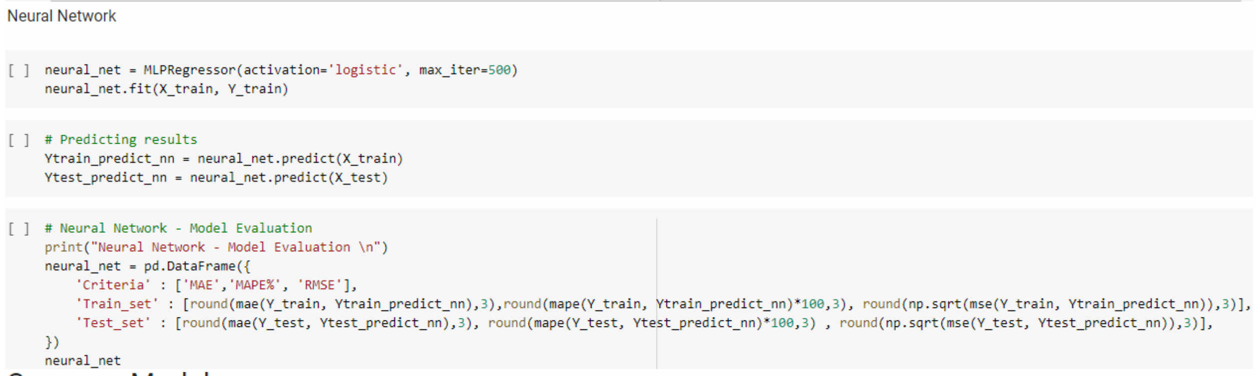


Random Forest Regression

```
[ ]  rf_regressor = RandomForestRegressor(n_estimators = 1000, min_samples_leaf = 2, max_features = 0.3)
     rf_regressor.fit(X_train,Y_train)
```

```
[ ]  # Predicting results
     Ytrain_predict_rf = rf_regressor.predict(X_train)
     Ytest_predict_rf = rf_regressor.predict(X_test)
```

```
[ ]  # Random Forest Regression - Model Evaluation
     print("Random Forest Regression - Model Evaluation \n")
     rf_reg = pd.DataFrame({
         'Criteria' : ['MAE','MAPE%', 'RMSE'],
         'Train_set' : [round(mae(Y_train, Ytrain_predict_rf),3),round(mape(Y_train, Ytrain_predict_rf)*100,3), round(np.sqrt(mse(Y_train, Ytrain_predict_rf)),3)],
         'Test_set' : [round(mae(Y_test, Ytest_predict_rf),3), round(mape(Y_test, Ytest_predict_rf)*100,3) , round(np.sqrt(mse(Y_test, Ytest_predict_rf)),3)],
     })
     rf_reg
```

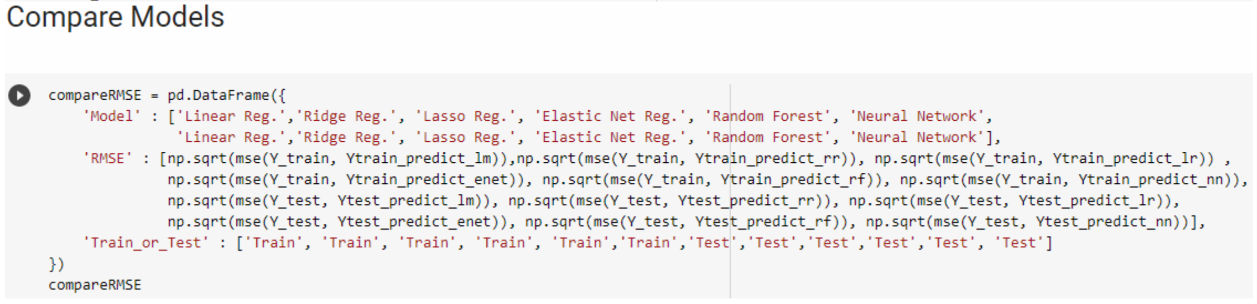


Neural Network

```
[ ]  neural_net = MLPRegressor(activation='logistic', max_iter=500)
     neural_net.fit(X_train, Y_train)
```

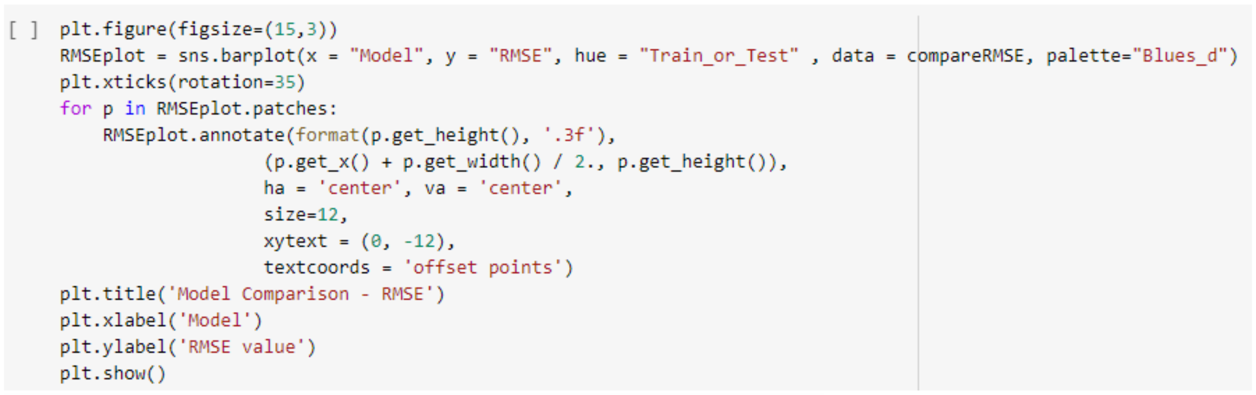
```
[ ]  # Predicting results
     Ytrain_predict_nn = neural_net.predict(X_train)
     Ytest_predict_nn = neural_net.predict(X_test)
```

```
[ ]  # Neural Network - Model Evaluation
     print("Neural Network - Model Evaluation \n")
     neural_net = pd.DataFrame({
         'Criteria' : ['MAE','MAPE%', 'RMSE'],
         'Train_set' : [round(mae(Y_train, Ytrain_predict_nn),3),round(mape(Y_train, Ytrain_predict_nn)*100,3), round(np.sqrt(mse(Y_train, Ytrain_predict_nn)),3)],
         'Test_set' : [round(mae(Y_test, Ytest_predict_nn),3), round(mape(Y_test, Ytest_predict_nn)*100,3) , round(np.sqrt(mse(Y_test, Ytest_predict_nn)),3)],
     })
     neural_net
```

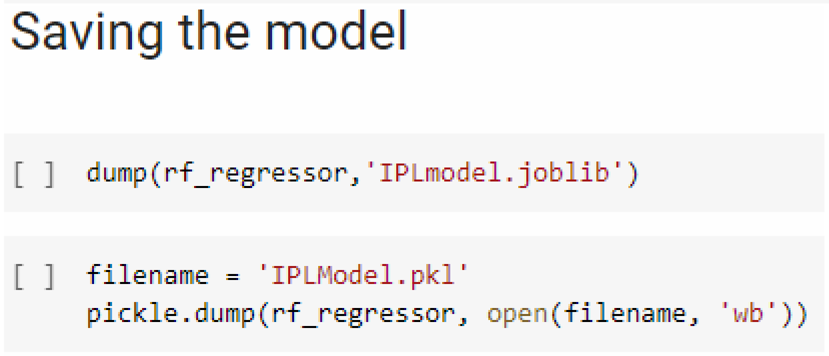


Compare Models

```
compareRMSE = pd.DataFrame({
    'Model' : ['Linear Reg.','Ridge Reg.', 'Lasso Reg.', 'Elastic Net Reg.', 'Random Forest', 'Neural Network',
               'Linear Reg.','Ridge Reg.', 'Lasso Reg.', 'Elastic Net Reg.', 'Random Forest', 'Neural Network'],
    'RMSE' : [np.sqrt(mse(Y_train, Ytrain_predict_lm)),np.sqrt(mse(Y_train, Ytrain_predict_rr)), np.sqrt(mse(Y_train, Ytrain_predict_lr)) ,
              np.sqrt(mse(Y_train, Ytrain_predict_enet)), np.sqrt(mse(Y_train, Ytrain_predict_rf)), np.sqrt(mse(Y_train, Ytrain_predict_nn)),
              np.sqrt(mse(Y_test, Ytest_predict_lm)), np.sqrt(mse(Y_test, Ytest_predict_rr)), np.sqrt(mse(Y_test, Ytest_predict_lr)),
              np.sqrt(mse(Y_test, Ytest_predict_enet)), np.sqrt(mse(Y_test, Ytest_predict_rf)), np.sqrt(mse(Y_test, Ytest_predict_nn))],
    'Train_or_Test' : ['Train', 'Train', 'Train', 'Train', 'Train','Train','Test','Test','Test','Test','Test', 'Test']
})
compareRMSE
```



```
[ ] plt.figure(figsize=(15,3))
    RMSEplot = sns.barplot(x = "Model", y = "RMSE", hue = "Train_or_Test" , data = compareRMSE, palette="Blues_d")
    plt.xticks(rotation=35)
    for p in RMSEplot.patches:
        RMSEplot.annotate(format(p.get_height(), '.3f'),
                       (p.get_x() + p.get_width() / 2., p.get_height()),
                       ha = 'center', va = 'center',
                       size=12,
                       xytext = (0, -12),
                       textcoords = 'offset points')
    plt.title('Model Comparison - RMSE')
    plt.xlabel('Model')
    plt.ylabel('RMSE value')
    plt.show()
```



Saving the model

```
[ ] dump(rf_regressor,'IPLmodel.joblib')
```

```
[ ] filename = 'IPLModel.pkl'
    pickle.dump(rf_regressor, open(filename, 'wb'))
```